A Minimal Method for Restoring Temporal Information Consistency

Mahat Khelfallah and Belaïd Benhamou

LSIS - CMI , Université de Provence
39 Rue Joliot-Curie - 13453 Marseille - France
{mahat,Belaid.Benhamou}@cmi.univ-mrs.fr

Abstract. Information often comes from different sources and merging these sources usually leads to the apparition of conflicts which have to be detected then eliminated in order to restore the information consistency. In this paper, we are interested in temporal information in the framework of simple temporal problems (STP). We propose a method which restores the consistency of an STP in a minimal way, i.e., by correcting a minimal number of constraints. This method computes the minimal subsets of constraints whose correction is sufficient to eliminate all the conflicts of the STP, without an exhaustive detection of these conflicts. The method we propose is based on Reiter's algorithm for computing the hitting sets of a collection to identify the minimal subsets of constraints to correct.
Keywords: Temporal constraints, STP, Restoring consistency.

1 Introduction

Information often comes from different sources and merging these sources usually leads to the apparition of conflicts which have to be detected then eliminated in order to restore the information consistency.

One of the famous formalisms for representing temporal information is simple temporal problem (STP) formalism proposed by Dechter, Meiri and Pearl [2] where temporal information is represented by linear constraint networks. Many researches investigated the STP resolution [2, 12]. However, the resolution methods do not propose any solution when the STP is inconsistent. Few research works focused on inconsistent STPs. We can cite [5] where a local method is proposed to merge STPs. This method considers the case where the merging operation of STPs leads to an inconsistent STP, and proposes a local method for restoring its consistency. However, this method does not restore the consistency in a minimal way. Indeed, it corrects more constraints than needed to eliminate all the conflicts.

In this paper, we are interested in restoring the consistency of temporal information in the framework of simple temporal problems (STP) in a minimal way. We propose a method which restores the consistency of an STP by correcting a minimal number of constraints. The classical idea for doing that consists first in detecting all the conflicts of the STP, then in identifying the conflicting constraints, and finally in correcting these constraints. The main drawback of this

© *A. Gelbukh, S. Torres, I. López (Eds.)*
Advances in Computer Science and Engineering
Research in Computing Science 19, 2006, pp. 43-56

idea is the exhaustive detection of conflicts of the STP which can be impossible when the STP is highly conflicting.

The method we propose here, computes the smallest subsets of constraints to correct without explicitly enumerating all the conflicts. It is based on Reiter's algorithm for computing the hitting sets of a set collection [8].

The rest of this paper is organized as follows. In section 2, we recall some background on simple temporal problems (STPs). The conflict detection algorithm is presented in section 3. We show in section 4 how to identify subsets of constraints to correct in order to restore the consistency. Section 5 describes how the correction operation is performed. In section 6, we give our consistency restoration method. In section 7, we present some previous related work and then we conclude in section 8.

2 Background

A *simple temporal problem* (STP) S is defined by $S=(X, C)$ where X is a finite set of variables $X_0, ..., X_n$, having continuous domains and representing temporal points. C is the set of constraints of the form $X_j - X_i \leq a_{ij}$ defined on these variables, where a_{ij} is a real scalar. Constraints of the form $X_j - X_i \geq a_{ij}$ can be also represented since $X_j - X_i \geq a_{ij}$ is equivalent to the constraint $X_i - X_j \leq -a_{ij}$. A tuple $X = (x_1, ..., x_n)$ of real values is a *solution* of the STP S if the instantiation $\{X_1{=}x_1, ..., X_n{=}x_n\}$ satisfies all its constraints. The STP S is *consistent* if and only if it has a solution.

The STP $S=(X, C)$ is associated with a directed edge-weighted graph, $G_d = (X, E_d)$, called the *distance graph* where X the set of vertices is the set of variables of the STP S, and E_d is the set of arcs representing the set of constraints C. Each constraint $X_j - X_i \leq a_{ij}$ of C is represented by the arc $i \rightarrow j$[1], which is weighted by a_{ij}. For more details see [2].

Example 1. We consider the STP $S = (X, C)$ where $X = \{X_0, X_1, X_2, X_3, X_4\}$ and

$$C = \left\{ \begin{array}{ll} X_0 - X_1 \leq -40\,, & X_2 - X_0 \leq -30\,, \\ X_3 - X_1 \leq 10\,, & X_2 - X_4 \leq -45\,, \\ X_4 - X_3 \leq 20\,, & X_1 - X_2 \leq 10\,, \\ X_2 - X_3 \leq -25\,, & X_0 - X_2 \leq 20 \end{array} \right\}$$

The STP S is associated with the distance graph shown in Figure 1. This example will be used in the next sections to illustrate our method.

3 Conflict Detection

In this section, we shall see how conflicts are detected in an inconsistent STP.

Let S be an inconsistent STP and G_d be the distance graph associated with the STP S.

[1] For simplicity a vertex X_i of the graph G_d is denoted by its index i.

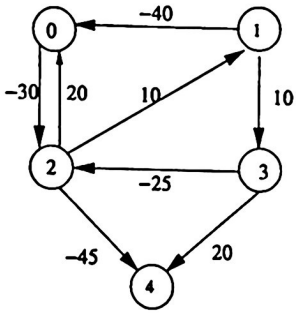

Fig. 1. The distance graph of the STP S defined in Example 1

The method that we propose to detect conflicts in the STP S is based on a well known result which we recall in the following theorem.

Theorem 1. *[9, 7, 6] An STP is consistent if and only if its corresponding distance graph does not contain negative circuits*[2].

We can deduce from Theorem 1 that for restoring the consistency of an STP, we need to remove all the negative circuits in its distance graph. We give a weaker condition for removing all the conflicts of an STP than the one stipulated in Theorem 1. It is sufficient to remove all the elementary[3] negative circuits in the distance graph to restore the consistency of an STP, as stated in the following theorem.

Theorem 2. *An STP is consistent if and only if its corresponding distance graph does not contain elementary negative circuits.*

Proof. ($\Rightarrow$) If an STP is consistent then its distance graph does not contain negative circuits (Theorem 1), and in particular it does not contain any elementary negative circuit. ($\Leftarrow$) If there is no elementary negative circuits in the distance graph, then it does not contain negative circuits (since a circuit is formed from elementary circuits), and by using Theorem 1, we conclude that the corresponding STP is consistent. □

The detection of conflicts amounts to detecting elementary negative circuits instead of detecting negative circuits. Thus, we can associate a conflict of S with an elementary negative circuit of G_d, a such conflict is defined as follows.

Definition 1. *Let S be an STP and G_d be its distance graph. A conflict of S is a pair (σ, d) where σ is an elementary negative circuit of the distance graph G_d and d is the distance*[4] *of the circuit σ.*

[2] A negative circuit is a circuit whose the sum of its arc weights is negative.
[3] An elementary circuit is a circuit which does not contain any smaller circuit.
[4] The distance of a path is the sum of its arc weights.

We recall that each arc $i \rightarrow j$ in G_d, weighted by a_{ij}, represents the constraint $c_{ij} : X_j - X_i \leq a_{ij}$ of S. We can define now the notion of conflicting constraint.

Definition 2. *Let $S = (X, C)$ be an STP and let $c = (\sigma, d)$ be a conflict of S. A constraint $c_{ij} \in C$ is a conflicting constraint of the conflict c if and only if $(i \rightarrow j)$ is an arc of the elementary negative circuit σ.*

Let c be a conflict of the STP S. We define $ConfConst(c)$ as the conflicting constraint set of the conflict c.

The number of conflicts in a STP is equal to the number of elementary negative circuits of its distance graph. This number can be exponential on the number of the STP variables. Therefore, an exhaustive conflict detection can be time and space consuming, and has to be avoid.

The *Conflict-Detection* function given by Algorithm 1 detects a conflict of the STP and returns the set of its conflicting constraints. This function is based on the *Floyd-Warshall* algorithm [3, 1]. The idea is to compute the shortest path for each pair of vertices (i, j) of the distance graph associated with the STP. In particular, if $i = j$, the algorithm computes the shortest circuit visiting the vertex i. If such circuit is negative, a conflict is detected and the algorithm returns the set of conflicting constraints involved in the detected negative elementary circuit. Otherwise, the algorithm returns the empty set.

Proposition 1. *Let S be an STP. The Conflict-Detection function applied to the STP S returns the set of conflicting constraints of a conflict of the STP S if S is inconsistent. Otherwise, it returns the empty set.*

Proof. The proof can be established trivially from the correction of the Floyd-Warshall algorithm [3].

Complexity of the Conflict-Detection function. The initialization step is done in $O(|X|^3)$. Testing if a path is elementary (line 8) can be done in $O(|X|)$, and the complexity of the function *ConfConst* which returns the conflicting constraints involved in the its argument is $O(|X|)$ (each elementary negative circuit of the distance graph can contain at most $|X|$ vertices, therefore it can involve at most $|X| - 1$ constraints). Hence, the complexity of the *Conflict-Detection* function is $O(|X|^4)$.

4 Identification of Minimal Subsets of Constraints to Correct

In order to guarantee the elimination of all the conflicts, at least one conflicting constraint of each conflict has to be corrected. In other words, the intersection of the set of corrected constraints and the set of conflicting constraints of each conflict has to be not empty. Let $\mathcal{L}_S$ be the collection of sets defined by $\mathcal{L}_S =$

[5] The $\bullet$ sign represents the path concatenation operation.

Algorithm 1 - Conflict-Detection$(X, C) : C'$

X - set of variables
C - set of constraints
C' - set of conflicting constraints of a conflict of the STP (X, C)
Var *mat* - $|X| \times |X|$ matrix of (path,distance) pairs
Begin
{ *Initialization* }
1. **For** $i, j := 1$ **to** $|X|$ **do**
2. **If** the constraint $(X_j - X_i \leq a_{ij}) \in C$ **then** $mat_{ij} := (i \rightarrow j, a_{ij})$;
3. **else if** $i = j$ **then** $mat_{ii} := (\emptyset, 0)$; **else** $mat_{ij} := (\emptyset, \infty)$; **end_if**;
4. **End_if**;
5. **End_for**;
{ *Computing shortest paths* }
6. **For** $k := 1$ **to** $|X|$ **do**
7. **For** $i, j := 1$ **to** $|X|$ **do**
8. **If** $(mat_{ik}.path \bullet {}^5mat_{kj}.path)$ is an elementary path
9. **and** $(mat_{ik}.distance + mat_{kj}.distance < mat_{ij}.distance)$ **then**
10. $mat_{ij}.distance := mat_{ik}.distance + mat_{kj}.distance$;
11. $mat_{ij}.path := mat_{ik}.path \bullet mat_{kj}.path$;
12. **If** $i = j$ **then return** $ConfConst(mat_{ii}.path)$; **end_if**;
13. **End_if**;
14. **End_for**;
15. **End_for**;
16. **Return** $\emptyset$;
End

$\{ConfConst(c) : c$ is a conflict of the STP $S\}$. Therefore, the subset of corrected constraints is a *hitting set* of the collection $\mathcal{L}_S$. The minimization of the corrected constraint number needs to find a minimal hitting set of the collection $\mathcal{L}_S$. We recall the definitions of a hitting set and a minimal hitting set.

Definition 3. *Let $\mathcal{L}$ be a collection of sets. H is a* hitting set *of the collection $\mathcal{L}$ if and only if for each set s of $\mathcal{L}$, $H \cap s \neq \emptyset$.*
A hitting set H_m of a collection $\mathcal{L}$ is minimal (according to cardinality) if and only if for each hitting set H of $\mathcal{L}$, $|H_m| \leq |H|$.

Now, we can present the *Minimal-Hitting-Sets* function (Algorithm 2), which computes the minimal hitting sets of the collection of sets representing the conflicts of an STP.

When the *Minimal-Hitting-Sets* algorithm is applied to the STP $S = (X, C)$, it returns the sets of all minimal hitting sets of the collection $\mathcal{L}_S$. It constructs a directed acyclic graph DAG for the collection $\mathcal{L}_S$. Each node n of this DAG is labeled by the set $Label(n)$ which is the set of the conflicting constraints involved in a conflict of the STP defined by $(X, C \setminus H(n))$ where $H(n)$ is the set of labels (which are constraints) of the branches from the DAG's root to the node n.

Algorithm 2 - Minimal-Hitting-Sets(S) : HS

$S = (X, C)$ - an inconsistent STP
HS - set of the minimal hitting sets of $\mathcal{L}_S$

Begin
1. $HS := \emptyset$;
2. Create a node n;
3. $Label(n) :=$ *Conflict-Detection*(X, C);
4. $H(n) := \emptyset$;
5. $lnc := \{n\}$;
6. **While** $lnc \neq \emptyset$ **do**
7. $lns := \emptyset$;
8. **For** each node $n \in lnc$ **do**
9. **For** each conflicting constraint $c \in Label(n)$ **do**
10. **If** there is a node $n' \in lns$ such that $H(n') = H(n) \cup \{c\}$ **then**
11. Point the branch leaving n and labeled by c to the node n';
12. **Else**
13. Construct the node n' the son of the node n;
14. $H(n') := H(n) \cup \{c\}$;
15. $Label(n') :=$ *Conflict-Detection*$(X, C \setminus H(n'))$;
16. **If** $Label(n') = \emptyset$ **then** $HS := HS \cup H(n')$; **end_if**;
17. $lns := lns \cup \{n'\}$;
18. **End_if**;
19. **End_for**;
20. **End_for**;
21. **If** $HS = \emptyset$ **then** $lnc := lns$; **else** $lnc := \emptyset$; **End_if**;
22. **End_while**;
23. **Return** HS;

End.

The initialization step of Algorithm 2 fixes the set of hitting sets HS to the empty set, and constructs the DAG's root n and labels it by the set of conflicting constraints of a conflict of the STP S. The function uses two node lists: lnc which is the current node list and lns which is the list of nodes which are successors of lnc nodes. lnc is set to the DAG's root n.

In the hitting sets computing step (lines 6-23), the DAG is constructed level by level until the list lnc becomes empty. In the "for" loop (lines 8-20), each node n of the lnc list has $|Label(n)|$ successors. The branch connecting the node n to one of its successors is labeled by a conflicting constraint $c \in Label(n)$. Before constructing a new successor of the node n, we check (line 10) if there is not a node n' which can be reused. If a such node exists then n' is a successor of n (line 11). Otherwise (lines 13-17), a new node n' is generated and is inserted into the node list lns. The label of the node n' is a detected conflict of the STP defined by $(X, C \setminus H(n'))$. The removal of the constraints $H(n')$ from the initial set of constraints C guarantees that $Label(n') \cap H(n') = \emptyset$. If $Label(n') = \emptyset$, i.e., if no conflict is detected then the set $H(n')$ is a minimal hitting set and is added

to the set HS (line 16). The construction of a new level of the DAG is stopped when a hitting set is computed ($HS \neq \emptyset$), in order to avoid non-minimal hitting set computing.

Let apply Algorithm 2 to our example.

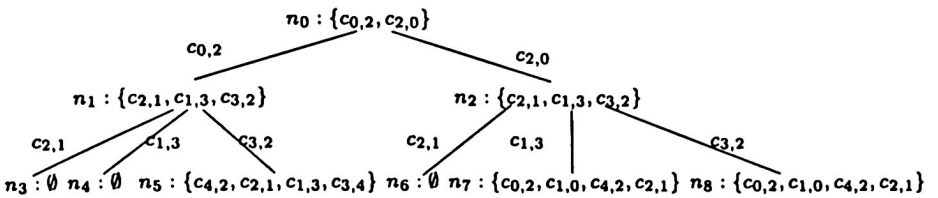

Fig. 2. Hitting sets search DAG for the STP defined in example 1

Example 2. The application of the minimal hitting sets function (Algorithm 2) to the STP defined in example 1 constructs the DAG depicted in figure 2. We can see that this DAG contains 3 nodes labeled by "$\emptyset$" which are nodes n_3, n_4 and n_6. This identifies the three following minimal hitting sets: $h_1 = \{c_{0,2}, c_{2,1}\}$, $h_2 = \{c_{0,2}, c_{1,3}\}$ and $h_3 = \{c_{2,0}, c_{2,1}\}$. Successors of the nodes n_5, n_7 and n_8 have not been constructed because every hitting set obtained after that will contain more than two constraints and therefore cannot be a minimal hitting set.

The following proposition states that the minimal hitting sets algorithm is correct and complete.

Proposition 2. *The Minimal-Hitting-Sets algorithm is correct and complete.*

Proof. To prove the correctness and completeness of the Minimal-Hitting-Sets algorithm, it is sufficient to prove the following three points: (1) the initial algorithm without pruning computes all the minimal hitting sets, (2) the pruning rules that we use do not eliminate some minimal hitting sets, and (3) the pruning rules eliminate all non-minimal hitting sets.

The access to the collection $\mathcal{L}_S$ is correct by Proposition 1 which states that the Conflict-Detection function returns an empty set if the input STP is consistent.

1. Proof by induction on the cardinality of the collection $\mathcal{L}_S$.
   Base: If $|\mathcal{L}_S| = 0$, then the set of hitting sets of $\mathcal{L}_S = \{\emptyset\}$, and the DAG associated with $\mathcal{L}_S$ contains only one node which is the root node n_0 labeled by $Label(n_0) = \emptyset$. As $H(n_0) = \emptyset$, Algorithm 2 computes all the minimal hitting sets of the collection $\mathcal{L}_S$.
   Induction: Let $l > 0$. (Induction hypothesis) Suppose that for every collection $\mathcal{L}_S$ such that $|\mathcal{L}_S| \leq l$, Algorithm 2 computes all the minimal hitting sets of $\mathcal{L}_S$. We prove that this is true also for collections $\mathcal{L}_S$ such that $|\mathcal{L}_S| > l$.

Let T be a DAG associated with the collection $\mathcal{L}_S$. Let n_0 be the T root. n_0 is labeled by a set $L_0 = \{m_1, \ldots, m_k\}$ such that $L_0 \in \mathcal{L}_S$. Let m_i be an element of L_0. Let $\mathcal{L}_{S_i}$ be the collection of sets which are elements of the collection $\mathcal{L}_S$ and which do not contain the element m_i, i.e., $\mathcal{L}_{S_i} = \{L \in \mathcal{L}_S : m_i \notin L\}$. The sub-DAG under m_i is a DAG associated with the collection $\mathcal{L}_{S_i}$. Since $|\mathcal{L}_{S_i}| < |\mathcal{L}_S|$, $|\mathcal{L}_{S_i}| \leq l$. Hence, by the induction hypothesis, the DAG associated with the collection $\mathcal{L}_{S_i}$ computes all the minimal hitting sets of $\mathcal{L}_{S_i}$. Thus every minimal hitting set of the collection $\mathcal{L}_{S_i}$ corresponds to a set of constraints H of a node labeled by "$\emptyset$" in the DAG associated with the collection $\mathcal{L}_{S_i}$. This implies that for every set of the form $h_i \cup \{m_i\}$ where h_i is a minimal hitting set of $\mathcal{L}_{S_i}$, there is a node n of the DAG T such that $H(n) = h_i \cup \{m_i\}$ and $Label(n) = \emptyset$. To prove the completeness of our algorithm, it is sufficient to prove that every minimal hitting set h of the collection $\mathcal{L}_S$ is such that $h = h_i \cup \{m_i\}$ where $m_i \in Label(n_0)$ and h_i is a minimal hitting set of the collection $\mathcal{L}_{S_i}$.

Let h be a minimal hitting set of $\mathcal{L}_S$. By definition, there is an element m_i such that $m_i \in h$ and $m_i \in L_0$ ($L_0 \in \mathcal{L}_S$). This element m_i is sufficient to remove all the sets $L_i \in \mathcal{L}_S$ such that $m_i \in L_i$. The remaining elements of h must belong to the remaining sets of the collection $\mathcal{L}_S$, i.e., $h \setminus \{m_i\}$ must be a minimal hitting set of the collection $\mathcal{L}_{S_i}$, which is true by construction.

2. We prove that the pruning rules we use do not remove nodes n such that $Label(n) = \emptyset$ and $H(n)$ is minimal. Let T be the DAG associated with the collection $\mathcal{L}_S$.

   (a) Reusing nodes (line 11 of Algorithm 2) does not remove nodes. It avoids the construction of nodes n' when there is a node n of T such that $H(n) = H(n')$. Let $\mathcal{L}_{S1}$ and $\mathcal{L}_{S2}$ be the collections defined by $\mathcal{L}_{S1} = \{L \in \mathcal{L}_S : L \cap H(n) \neq \emptyset\}$ and $\mathcal{L}_{S2} = \{L \in \mathcal{L}_S : L \cap H(n) = \emptyset\}$. Suppose that T_1 and T_2 are the sub-DAGs of T, without pruning, having as root respectively nodes n and n'. We prove that the minimal hitting sets obtained from the sub-DAG T_2 are also obtained from T_1. Suppose that there is a node n'_l of T_2 such that $Label(n'_l) = \emptyset$ and $H(n'_l) = h$ where h is minimal hitting set of $\mathcal{L}_S$. Actually, $h = H(n') \cup h'$ where h' is a minimal hitting set of $\mathcal{L}_{S2}$ because $H(n')$ is a minimal hitting set of $\mathcal{L}_{S1}$. Since h' is a minimal hitting set of $\mathcal{L}_{S2}$ and T_1 is the DAG associated with the collection $\mathcal{L}_{S2}$, there is a node n_l of T_1 such that $Label(n_l) = \emptyset$ and $H(n_l) = H(n) \cup h'$ (the set of branch labels from the node n to the node n_l is the minimal hitting set h'). However, $H(n) = H(n')$, thus there is a node n_l of T_1 such that $Label(n_l) = \emptyset$ and $H(n_l) = h$. Therefore, T_1 and T_2 compute the same minimal hitting sets.

   (b) To prove that stopping the construction of more levels of the DAG T after the generation of a node n such that $Label(n) = \emptyset$ does not eliminate minimal hitting sets of the collection $\mathcal{L}_S$, it is sufficient to prove that the obtained hitting sets from non developed levels are not minimal. Let i be the smallest number such that there is a node n of T where $Label(n) = \emptyset$ and $|H(n)| = i$. Let n' be a node of T such that $Label(n') = \emptyset$ and $|H(n')| > i$. $H(n)$ and $H(n')$ are hitting sets of $\mathcal{L}_S$. But, $|H(n')| > H(n)$, thus, $H(n')$ is not a minimal hitting set of $\mathcal{L}_S$.

3. Suppose there is a node n of the non-pruned DAG such that $Label(n) = \emptyset$ and $H(n)$ is not a minimal hitting set of $\mathcal{L}_S$. Let h_m be a minimal hitting set of $\mathcal{L}_S$. From point 2 of this proof, we deduce that there is a node n' such that $Label(n') = \emptyset$ and $H(n') = h_m$. As $H(n)$ is not a minimal hitting set of $\mathcal{L}_S$ and $H(n')$ is a minimal one, thus by definition, $|H(n')| < |H(n)|$. This means that the level of node n is under n''s one. However, the construction of DAG levels is stopped after the generation of a node labeled by "$\emptyset$". Therefore, the node n will not be constructed. □

Proposition 3. *Let $S = (X, C)$ be an inconsistent STP and $\mathcal{L}_S$ be the collection of sets defined by $\mathcal{L}_S = \{\ ConfConst(c) : c$ is a conflict of $S\}$. The maximal depth of the DAG of minimal hitting sets search of the collection of $\mathcal{L}_S$ is $|C|$.*

Proof. Let p be the depth of the DAG T constructed for minimal hitting set search of the collection $\mathcal{L}_S$. Let h be a minimal hitting set of $\mathcal{L}_S$. Thus, there is a node n of T such that $Label(n) = \emptyset$ and $H(n) = h$ (correction of the Minimal-Hitting-Sets algorithm, by Proposition 2). Since all the minimal hitting sets of $\mathcal{L}_S$ have the same cardinality, $p = |H(n)|$, and thus $p = |h|$. By definition, $\mathcal{L}_S$ elements are subsets of constraints, thus $h \subseteq C$. Therefore, $|h| \leq |C|$, and $p \leq |C|$. □

Complexity of the Minimal-Hitting-Sets algorithm. Let X and C be respectively the set of variables and the set constraints of the input STP S. Conflict detection (line 3) can be done in $O(|X|^4)$ and the other initialization operations are elementary and can be done is constant time. At every iteration of the loop (6-23), a level of the DAG is constructed. The maximal depth of this DAG is $|C|$ (Proposition 3), then there is at most $|C|$ iterations of the loop (6-23). We compute now the complexity of each of its iterations. The loop (8-20) performs $|lnc|$ iterations. For every node n of the list lnc, the loop (9-19) performs $|Label(n)|$ iterations. Since the set $Label(n)$ contains at most $|X|$ constraints, then the number of iterations of the loop (9-19) is bounded by $|X|$. In line 10, $H(n)$ is compared with the set H of lns elements. Each comparison can be done in $O(|C|)$ because H contains at most $|C|$ constraints. Thus, the test is performed in $O(|C| \times |lns|)$. The complexity of conflict detection is $O(|X|^4)$, thus the complexity of lines (12-18) is $O(|X|^4)$. Therefore, the complexity of the loop (9-19) is $O(|X| \times (|X|^4 + |C| \times |lns|))$. The loop (8-20) performs $|lnc|$ iterations, thus its complexity is $O(|lnc| \times |X| \times (|X|^4 + |C| \times |lns|))$. Now, we compute the bounds of $|lnc|$ and $|lnc|$. At iteration i of the loop (8-20), lnc contains nodes created at the iteration $i-1$. Their number is bounded by the number of subsets of constraints whose cardinality is equal to i. This number is roughly bounded by $2^{|C|}$ which is the number of all the subsets of C. Similarly, $|lns|$ is bounded by $2^{|C|}$. Therefore, the complexity of the loop (8-20) is in $O(|X| \times |C| \times (2^{|C|})^2)$. Since the loop (6-23) performs at most $|C|$ iterations, then its complexity is in $O(|C|^2 \times |X| \times (2^{|C|})^2)$, and the complexity of the *Minimal-Hitting-Sets* algorithm is $O(|C|^2 \times |X| \times 2^{2 \times |C|})$.

We prove in the next section that the correction of the constraints corresponding to a minimal hitting set of the collection $\mathcal{L}_S$ corresponds to a minimal consistency restoration of the STP S.

5 Correction of the Conflicting Constraints

Now, we shall see how to perform the corrections. Let $c = (\sigma, d)$ be a conflict of the STP S. The elimination of the conflict c needs the elimination of its associated elementary negative circuit σ. This implies the correction of at least one of the constraints involved in σ, i.e., one of the constraints of $ConfConst(c)$. The following proposition shows how this correction is made.

Proposition 4. *Let S be an STP and $c = (\sigma, d)$ be a conflict of S. Let $c_{ij} : X_j - X_i \leq a_{ij}$ be a conflicting constraint of c, i.e., $c_{ij} \in ConfConst(c)$. Replacing the constraint $c_{ij} : X_j - X_i \leq a_{ij}$ by the constraint $X_j - X_i \leq a_{ij} - d$ eliminates the conflict c.*

Proof. Let $c = (\sigma, d)$ be a conflict, and let $c_{ij} : X_j - X_i \leq a_{ij}$ be a conflicting constraint of c, $c_{ij} \in ConfConst(c)$. The circuit σ has a negative distance d, and replacing the constraint $X_j - X_i \leq a_{ij}$ by the constraint $X_j - X_i \leq a_{ij} - d$, will make this distance equal to zero. Hence the negative circuit σ is eliminated and the conflict c is corrected. □

Example 3. In the STP of Example 1, the elementary negative circuit $(0 \rightarrow 2 \rightarrow 0)$ whose distance is -10 is detected. This defines the conflict $((0 \rightarrow 2 \rightarrow 0), -10)$ between the constraints $X_2 - X_0 \leq -30$, $X_0 - X_2 \leq 20$. This conflict can be removed by either replacing the constraint $X_2 - X_0 \leq -30$ by the constraint $X_2 - X_0 \leq -20$, or replacing the constraint $X_0 - X_2 \leq 20$ by the constraint $X_0 - X_2 \leq 30$.

The following proposition states that a conflict is eliminated if and only if one of its conflicting constraints is corrected.

Proposition 5. *Let S be an STP and $c = (\sigma, d)$ be a conflict of S. The conflict c is corrected if and only if at least one of the constraints $c_{ij} \in ConfConst(c)$ is corrected with respect to Proposition 4.*

Proof. Let S be an STP and $c = (\sigma, d)$ be a conflict of S. ($\Rightarrow$) Suppose that the conflict $c = (\sigma, d)$ is eliminated. This implies that the negative circuit σ is eliminated, i.e., its distance is changed from negative to positive. This is done by changing at least the weight of one of its arcs. In other words, by correcting, at least, one constraint c_{ij} of $ConfConst(c)$. ($\Leftarrow$) The correction of a constraint $c_{ij} \in ConfConst(c)$ (with respect to Proposition 4) makes the conflict circuit σ positive. Hence, the conflict $c = (\sigma, d)$ is eliminated. □

Proposition 6 guarantees that when correcting a constraint no new conflicts are generated.

Proposition 6. *The correction of a constraint cannot generate new conflicts.*

Proof. Correcting a constraint does not decrease the distance of any circuit, then it cannot generate new negative circuits. Therefore, no new conflicts are generated. □

Now, we give the *Constraint-Correction* function (Algorithm 3) which corrects the subset of constraints C' given as input. First, this function computes the quantities to add to the corrected constraints for eliminating the conflicts. These quantities correspond to the distance of the shortest elementary negative circuits of the distance graph associated with the STP $S = (X, C)$. We apply the *Floyd-Warshall* algorithm [3, 1] to compute them.

Algorithm 3 - Constraint-Correction$(X, C, C') : C''$

X - set of variables
C - set of constraints
C' - set of constraints to correct
C'' - set of corrected constraints
Var : mat - $|X| \times |X|$ matrix of reals
Begin
1. **For** $i, j := 1$ **to** $|X|$ **do**
2. **If** there is a constraint $X_i - X_j \leq a_{ij}$ of C **then** $mat_{ij} := a_{ij}$;
3. **Else if** $i \neq j$ **then** $mat_{ij} := \infty$; **else** $mat_{ij} := 0$; **end_if**;
4. **End_if**;
5. **End_for**;
6. **For** $k := 1$ **to** $|X|$ **do**
7. **For** $i, j := 1$ **to** $|X|$ **do**
8. **If** $mat_{ij} > mat_{ik} + mat_{kj}$ **then** $mat_{ij} := mat_{ik} + mat_{kj}$; **end_if**;
9. **End_for**;
10. **End_for**;
11. $C'' := C$;
12. **For every** constraint $c_{ij} : X_j - X_i \leq a_{ij}$ of C' **do**
13. $C'' := C'' \setminus \{c_{ij}\} \cup \{c'_{ij} : X_j - X_i \leq a_{ij} - mat_{ij}\}$;
14. **End_for**;
15. **Return** C'';
End.

Proposition 7. *Let $S = (X, C)$ be an inconsistent STP and let C' be a subset of C. Applying the Constraint-Correction algorithm to respectively X, C and C' eliminates every conflict of S involving a conflicting constraint of C'.*

Proof. At the end of line 10, for every pair of variables (X_i, X_j) of X, $mat_{ij} \leq min_{ij}$, where min_{ij} is the distance of the shortest path from i to j in the distance graph of the STP S (correction of the Floyd-Warshall algorithm - see [3]). The rest of the proof is based on the proposition 4. □

Complexity of the Constraint-Correction function. The complexity of the loop (1-5) is $O(|X|^2)$, and the loop (6-10) is in $O(|X|^3)$. The loop (12-14) can be done in $O(|C|)$. Since $|C| < |X|^2$, then the complexity of the *Constraint-Correction* function is $O(|X|^3)$.

The following theorem states that the correction of the constraints corresponding to a minimal hitting set of the collection representing the conflicts of the STP is a minimal consistency restoration of the STP.

Theorem 3. *Let S be an STP and let $\mathcal{L}_S$ be the collection of sets representing the conflicts of S such that $\mathcal{L}_S = \{ConfConst(c) : c \text{ is a conflict of } S\}$. A minimal consistency restoration of S is equivalent to the correction of the constraints corresponding to a minimal hitting set of the collection $\mathcal{L}_S$.*

Proof. ($\Rightarrow$) Let C_m be a minimal subset of corrected constraints in a minimal restoration of consistency of the STP S. Each conflict c of S is removed by the correction of, at least, one constraint c_{ij} of $ConfConst(c)$ (Proposition 5). This means that $C_m \cap ConfConst(c) \neq \emptyset$ for every conflict c of S. $ConfConst(c)$ is also an element of the collection $\mathcal{L}_S$. Thus, for every $e \in \mathcal{L}_S$, $C_m \cap e \neq \emptyset$. Therefore, C_m is a hitting set of the collection $\mathcal{L}_S$. Furthermore, this hitting set is minimal because the consistency restoration corrects a minimal number of constraints.
($\Leftarrow$) Let h_m be a minimal hitting set of the collection $\mathcal{L}_S$, and suppose that the corresponding constraints are corrected. However, each element $e \in \mathcal{L}_S$ represents the conflicting constraint set of a conflict c of S, i.e., $e = ConfConst(c)$, and $e \cap h_m \neq \emptyset$. Then, we conclude that for each conflict c, at least one of its conflicting constraints is corrected. Hence, each conflict c is eliminated (Proposition 5). This implies that the correction of the constraints corresponding to h_m eliminates all the conflicts of S. Since h_m is minimal, we can conclude that the number of corrected constraints is minimal. □

6 STP Minimal Consistency Restoration Method

Now, we can describe our method for minimal consistency restoration for STP which is based on the minimal hitting sets search. This method is called *Min-Consistency-Restoration* and is given in Algorithm 4. The first step of this method is the minimal hitting set search of the collection $\mathcal{L}_S$ representing the conflicts of the input STP S. This operation is performed by applying the *Minimal-Hitting-Set* function (Algorithm 2). Each obtained minimal hitting set corresponds to a minimal (in term of cardinality) subset of constraints whose correction is sufficient to remove all the conflicts of the STP S. For every computed hitting set hs a consistent STP is obtained by the correction of the constraints corresponding to hs, by applying the *Constraint-Correction* function (Algorithm 3). The *Min-Consistency-Restoration* returns Σ a set of consistent STPs.

Theorem 4. *The Min-Consistency-Restoration algorithm is correct and complete.*

Proof. The proof is trivially established from Theorem 3 and Proposition 7. □

Algorithm 4 - Min-Consistency-Restoration(S) : Σ

S - STP
Σ - set of consistent STPs
Begin
1. $\Sigma = \emptyset$;
2. HS := *Minimal-Hitting-Sets*(S);
3. **For** every minimal hitting set hs of HS **do**
4. $\quad \Sigma = \Sigma \cup (X, Constraint\text{-}Correction\ (X, C, hs))$;
5. **End_for**;
6. **Return** Σ;

End

Complexity of the Min-Consistency-Restoration function. The complexity of the *Minimal-Hitting-Sets* function is $O(|C|^2 \times |X| \times 2^{2\times|C|})$. Since the number of minimal hitting sets $|HS| < 2^{|C|}$, the loop (3-5) performs $|HS| < 2^{|C|}$ iterations. Each iteration is done in $O(|X|^3)$. Therefore, the complexity of the *Min-Consistency-Restoration* function is $O(|C|^2 \times |X| \times 2^{2\times|C|})$.

7 Related Work

The Reiter's algorithm for diagnosis [8] has been adapted in previous works [11, 10] for revision methods in the framework of propositional logic. Our interest is to do that in the framework of constraints.

Some previous works investigated the consistency restoration of temporal information in the framework of STPs. Different revision strategies for a geographic application represented by an STP were proposed in [4]. One of them restores the consistency in a minimal way. The difference with our minimal method is that the STPs handled in [4] are particular STPs where the detection of all conflicts is done in polynomial time complexity. Our method deals with more general STPs where the number of conflicts can be exponential on the number of STP variables.

A local method for restoring the consistency of general STPs was proposed in [5]. This method returns a consistent STP but the number of corrected constraints is not minimized as in the method which we propose here.

8 Conclusion

In this paper, we focused on inconsistent simple temporal problems for which we proposed a minimal consistency restoration method. The classical idea to eliminate all the conflicts of an STP consists in detecting all these conflicts then in eliminating them. However, this idea can lead to time consuming method since the number of conflicts in an STP is in general exponential on the number of STP variables. The method we proposed avoids the exhaustive conflict detection,

and identifies the smallest subsets of constraints whose correction is sufficient to eliminate all the conflicts and to restore the consistency of the STP. This is achieved by searching the minimal hitting sets of the collection of sets whose elements are sets containing the conflicting constraints of a conflict of the STP S. The algorithm we propose to search hitting sets is an adaptation of the Reiter's algorithm [8] for diagnosis. We show that our consistency restoration method is correct and complete, i.e., it returns all the consistent STPs obtained by correcting a minimal number of constraints of the initial STP S.

In a future work, we are looking to investigate temporal constraints with preferences. We hope handle both quantitative and qualitative preferences. We are also, interested in fusion of disjunctive temporal problems with and without preferences.

References

1. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Massachusetts, 1990.
2. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
3. M. Gondron and M. Minoux. *Graphes et Algorithmes.* Eyrolles, 1979.
4. M. Khelfallah and B. Benhamou. Geographic information revision based on constraints. In *Proc. of the 14th European Conference on Artificial Intelligence (ECAI'04)*, pages 828–832, 2004.
5. M. Khelfallah and B. Benhamou. A local fusion method of temporal information. In *Proc. of the 8th European Conference on Symbolic and Qualitative Approaches to Reasoning with Uncertainty (ECSQARU'05)*, pages 477–488, 2005.
6. C.E. Leiserson and J.B. Saxe. A mixed-integer linear programming problem which is efficiently solvable. In *Proc. of the 21st annual Allerton conference on Communications, Control, and Computing*, pages 204–213, 1983.
7. Y.Z. Lia and C.K. Wong. An algorithm to compact a VLSI symbolic layout with mixed constraints. In *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, volume 2, pages 62–69, 1983.
8. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
9. R. Shostak. Deciding linear inequalities by computing loop residues. *Journal of ACM*, 28(4):769–779, 1981.
10. R. Wassermann. An algorithm for belief revision. In *Proc. of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, pages 345–352, 2000.
11. E. Würbel, R. Jeansoulin, and O. Papini. Revision: an application in the framework of GIS. In *Proc. of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, pages 505–516, 2000.
12. L. Xu and B. Choueiry. A new efficient algorithm for solving the simple temporal problem. In *10th Int. Symposium on Temporal Representation and Reasoning and 4th Int. Conf. on Temporal Logic (TIME-ICTL 03)*, pages 212–222, 2003.